Smart Home Environmental Monitoring System

AUTHOR 1: IZABELA ZELEK

AUTHOR 2: CHRIS LEIVON

DATE OF SUBMISSION: 11/12/2024

Executive Summary

This project develops a smart home safety system that uses various sensors—temperature, humidity, gas, and smoke—to monitor environmental conditions. Utilizing a Raspberry Pi for local data processing, the system employs Random Forests to predict the ideal environment for each user based on past choices. Key challenges include maintaining sensor accuracy and optimizing lightweight algorithms for edge computing. Designed for privacy, the system processes data locally. sends the data by an encrypted protocol MQTT to the server, and features an intuitive user interface for easy access, particularly for vulnerable populations. The user interface comes in the form of a webapp which displays the current home conditions, all alerts detected since initial setup of product, graphs showing average conditions each month according to historical data, and conditions set by the user themselves. The webapp allows the user to opt-out of collecting time and date as data to further increase the safety of the user. The sensor displays animated pictures upon detecting a hazardous situation such as a fire or gas leak, improving the usability for the elderly. The app also sends email alerts upon detecting such safety hazards to allow users to monitor their home while away. The project aims to enhance home safety and comfort through cost-effective and energy-efficient solutions.

Table of Contents

Introduction

Artificial intelligence is an ever-growing field which is becoming more and more prominent in every aspect of our everyday lives. As the artificial intelligence sector grows, more home inventions are created in order to increase the quality of living and perform menial household tasks for us. As a result, homes are becoming smarter and more technological, with particular focus being put on home safety. This project will utilise ubiquitous computing and machine learning in order to combine real-time monitoring with smart home automation.

The aim of this project is to design and develop a system for smart homes which monitors the environment using a variety of sensors, such as temperature, humidity, pressure and light sensors. The project will be created on the Raspberry Pi and will use the Sense HAT sensors to collect environment data. This system will detect harmful conditions, including gas leaks, fire risks, or unhealthy temperature levels. Sensor data will be processed locally on the Raspberry Pi, where machine learning algorithms, such as Decision Trees, will be implemented to make predictions in real time and adapt to changes in the environment. This project intends to address home safety issues by alerting the occupants of potential hazards such as gas leaks or fire by displaying a flashing warning on the LED. In addition to sensor warnings, the system will also send an alert through email to ensure the user stays informed about the condition of their home. Challenges may include optimizing the system for energy efficiency and ensuring that alerts are timely and accurate. Ultimately, the goal is to create a system that enhances home safety by combining real-time monitoring with smart home automation.

Objectives

Environmental Monitoring: Collect real-world data using the Sense Hat.

Database Management: Store all collected data in a database and allow the user to generate graphs based on the historical data.

Machine Learning: Use machine learning to adjust the home conditions based on learned user preferences.

Calibration: Re-calibrate the temperature and pressure sensors in order to maintain accuracy of readings.

User Interface: Create a webapp which takes data from the database and displays it.

Colour-Coded LED Alerts: Design and Program the LED to display an animation based on the detected safety hazard.

Email Alerts: Send email to the user whenever a safety hazard is detected in order to allow the user to stay informed.

Scope

This project includes:

- Sensors that detect the temperature. Pressure and humidity.
- A database to store the collected data both from the sensors and from the user's preferences.

- An MQTT Server to allow subscribers and publishers to send information.
- Alerts displaying on the LED, displaying on the webapp, and being sent to the user's email.
- A webapp to display information to the user with the ability to change users.
- An AI which calculates user's ideal home environment.

This project does not include:

- A fire, gas or light sensor, the data is instead determined from predefined conditions.
- A connection to heating systems, the changes are instead simulated on the webapp.

Significance

This application offers a real-time monitoring of their home, even while away. All data is stored in a database which allows for historical data to be taken which allows the user to understand condition trends in their house. This can help the user better manage their energy consumption by visualising what months require heating and more light. The app offers peace of mind to users by allowing them constant home monitoring. It also offers a design which can be combined with heating, dehumidifiers and other such appliances to allow the user to keep their home in perfect condition.

System Design

Overview of System Architecture

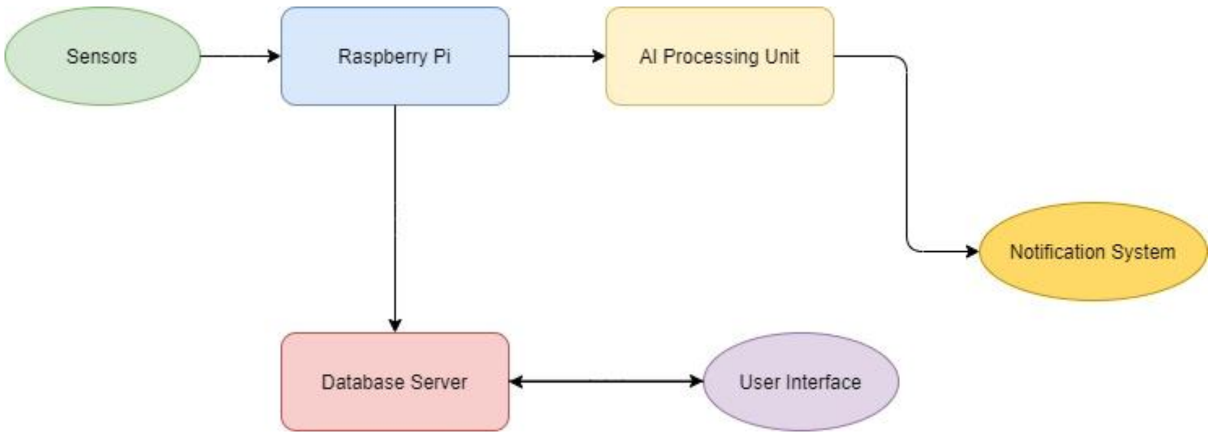


Figure 1: Overview of System Architecture Diagram

The architecture was based around the chart shown in Fig. 1. Temperature, Pressure and Humidity are measured in real-time using Sense HAT sensors. Due to a lack of gas, fire and light sensor, these values are decided based on predefined conditions. The sensors continuously collect environmental data from the home, which is then processed locally on the Raspberry Pi to ensure context-awareness. This ensures that the system minimizes false alerts by calibrating the sensors before sending notifications to the user. The data is then passed through a MQTT protocol to the server where it is stored in the database for future use.

Several machine learning algorithms, including Decision Trees, Tensorflow Lite, XGBoost, KNN, SVM, and Random Forest, were tested to determine which provides the most accurate predictions. Once Random Forest was determined as the most efficient algorithm, it was deployed on the Raspberry Pi to predict the ideal home conditions of each user based on historical data stored in the database. Edge processing on the Raspberry Pi allows the system to make real-time decisions on hazardous conditions, such as fires, before an alert is sent to the system to alert the user. The sensor itself displays a flashing animation upon detection of a fire or a gas leak, which ensures that even with network issues, the user is still notified.

User interaction is central to the system's design. An RGB LED visually indicates hazardous conditions, changing colour depending on the type of threat detected, such as red for fire, and green for gas leaks. The LED flashes specific images, such as a fire icon for flames or green floating gas for gas leaks. In addition to visual cues, an intuitive interface was developed for users to check the current status of the home and receive real-time alerts. The interface was integrated into a web dashboard, allowing the user to monitor conditions remotely.

The system also takes steps to prevent wrong readings. For example, if the temperature sensor detects the temperature, it recalibrates to ensure the reading does not take into account the heat off the sensor before sending the data. The system does not only warn the user but also offers the potential to be linked with integrated home thermostats and humidifiers. This functionality extends to light intensity regulation, allowing users to set preferences for how bright or dim their rooms are, and how warm they prefer their household.

Sensor Integration

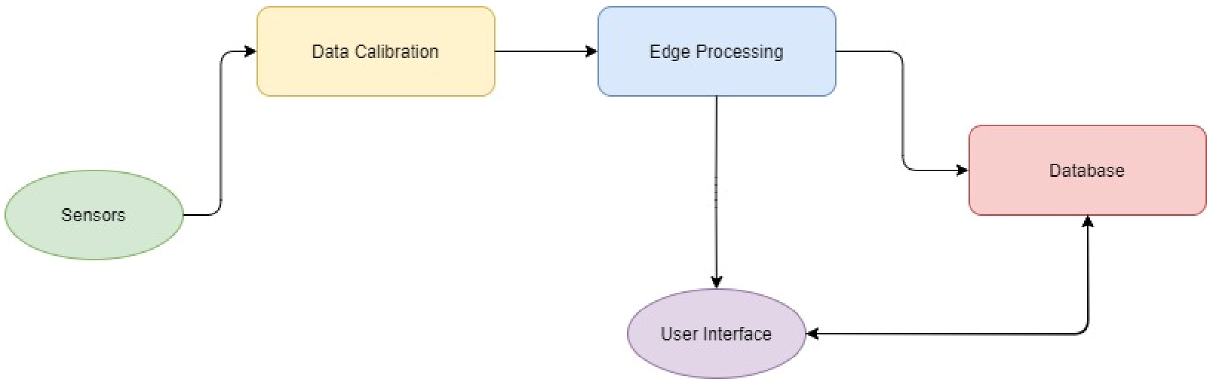


Figure 2: Sensor Integration Diagram

This project was created by following the chart shown in Fig. 2. It utilises the Sense HAT which is an add-on for Raspberry Pi. It contains various environmental sensors. The sensors used in this project were the temperature sensor, humidity sensor and the pressure sensor. The temperature sensor measures the temperature of the environment, out additional code recalculates the temperature to not consider the temperature of the Raspberry Pi itself. The humidity sensor detects the humidity of the environment. Finally, the pressure sensor detects atmospheric pressure.

All data is stored in the database, currently with three defined users: John, Jane and Default. The Default data is taken from the edge device which contains the sensors. Additional data, which can later be taken from additional sensors, is currently decided based on predefined conditions depending on the current month and time. This additional data consists of light intensity, to allow the user to determine the brightness or dimness of the house. Data for John and Jane are taken from the webapp which allows the users to set their preferred conditions. Random Forest is then used to learn these preferences to allow for automatic regulation. This data is also displayed on the webapp so that the user can monitor the changes in the household. The data tab on the left allows the user to see what data was collected and gives the option for the user to opt-out of time and date data collection, in the event the user is afraid of the information being potentially hazardous due to hackers being able to see the trends of the household.

AI and Edge Computing Techniques

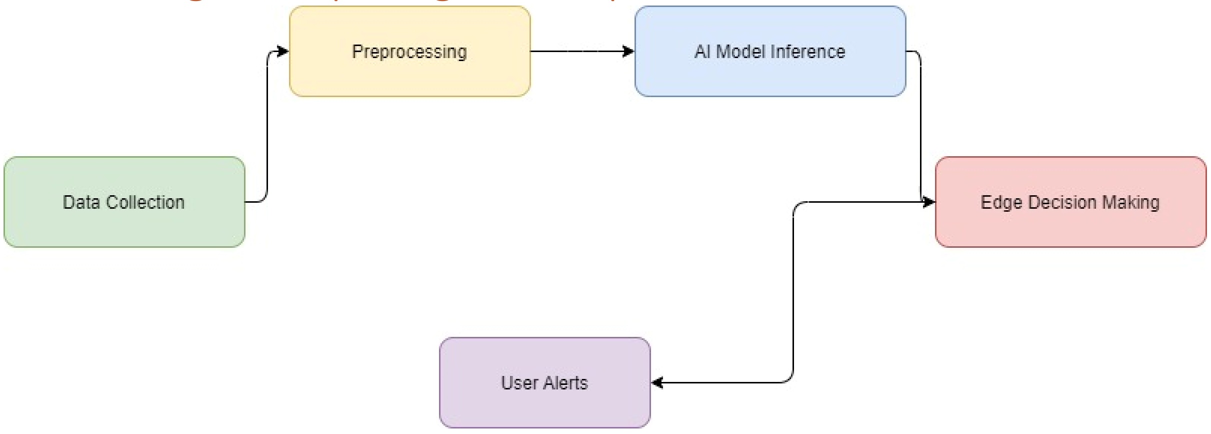


Figure 3: AI and Edge Computing Techniques Diagram

	Temperature		Light Intensity	
Models	**Mean Squared Error**	**R2 Score**	**Mean Squared Error**	**R2 Score**
SVM	5.97	0.68	17019.76	0.51
KNN	13.99	0.25	21918.85	0.36
XGBoost	2.23	0.88	5170.33	0.85
Random Forest	2.33	0.88	5114.60	0.85
Decision Tree	4.06	0.78	9062.15	0.74
TensorFlow Lite	16.69	0.11	16620.18	0.52

Figure 4: Comparison of Model Performance

AI Models Implemented: The structure of the ai and edge computing was based on Fig. 3. The AI algorithm/model used in this project is Random Forest. It is a powerful ensemble method which is based on Decision Tree. It accumulates predictions from multiple decision trees to increase its accuracy and reduce the likelihood of overfitting (GeeksforGeeks, 2024). In this project, Random Forest is used to predict the user's preferred condition based on the data collected over time from sensors and different preferences based on the time of day and month. Random Forrest was chosen after comparing the result with other machine learning algorithms, such as Decision Tree, Fuzzy Logic and some other algorithms, mainly due to its superior performance in handling non-linear relationships and the ability to work better with small datasets, which is very common in edge devices or environment, as shown in Fig. 4.

Edge Processing: Edge processing is very important in this project mainly for real-time hazard detection and environment monitoring. The Pi serves as the edge device, hosting the sensors. For data collection, data such as temperature, humidity, pressure and such is captured using the Sense-Hat sensors. Data for gas and fire data had to be simulated and are incorporated using predefined conditions. Sensor readings are being calibrated on the Pi to ensure accurate reading. For example, the temperature is recalibrated to exclude the heat that is generated by the Raspberry Pi's CPU. Using the trained algorithm, the Raspberry Pi predicts the user's ideal preference. The predictions are done locally on the Pi, minimizing latency and enabling fast and immediate responses to dangerous conditions through emails. The system also processes potential dangers regarding fire and gas leaks locally. If any threats are detected, the LED on the Raspberry Pi will display animation indicating the type of threat and emails will be sent to the user as well. This ensures reliability even when there is no network connection.

Optimization Strategies: To meet computational limitations on the Raspberry Pi, some optimization strategies were needed to be implemented. One of them is through using lightweight models like Random Forest. Features which were absolutely necessary like user's preferences, time and sensor data were included. Also, data processing steps, such as recalibration are excluded directly on the system to minimize the amount of data being processed by the algorithm. Sensor readings are taken at 15-minute intervals to balance real-time awareness with energy efficiency, which reduces unnecessary load and power consumption. By processing data locally, the system reduces dependency on server. This decreases latency and guarantees functionality during network outages. The personal data displayed on 'Data' page is limited to a hundred records per page to minimize amount of data shown and improve performance.

Communication Between Devices

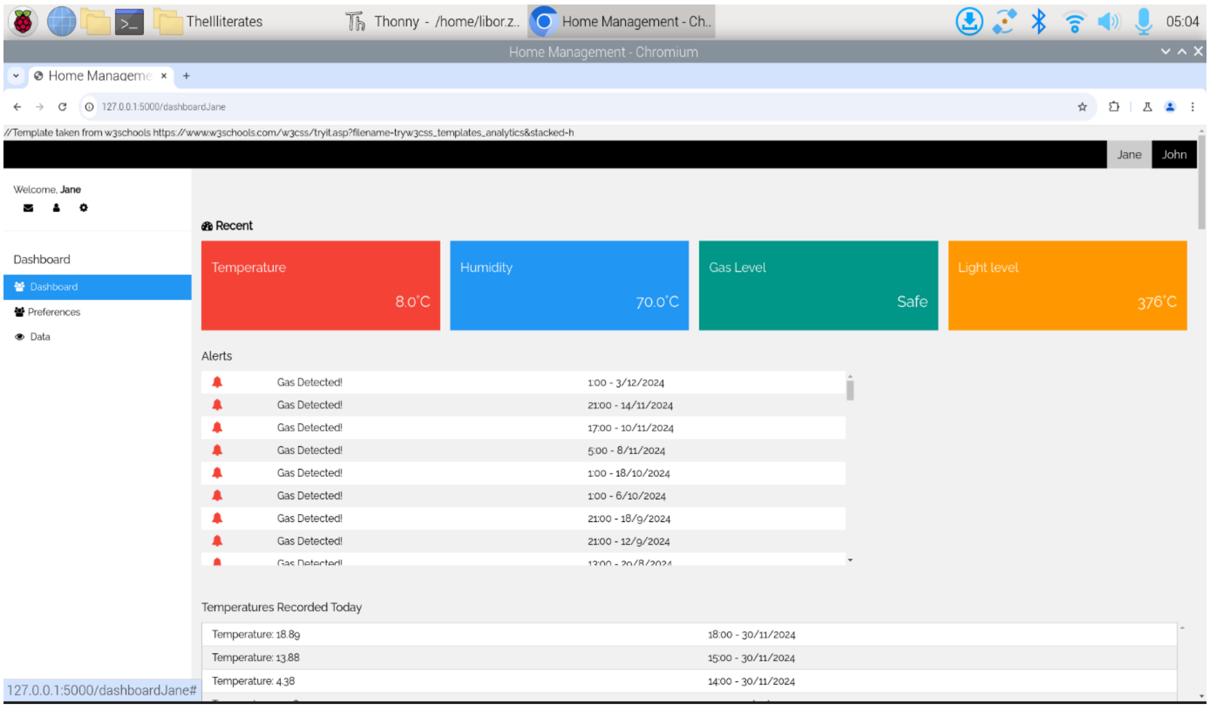
In this smart home safety system, devices communicate over a Wi-Fi network using the Message Queuing Telemetry Transport (MQTT) protocol. MQTT is a lightweight, publish-subscribe messaging protocol ideal for resource-constrained devices and low-bandwidth networks, making it well-suited for Internet of Things (IoT) applications (Tutorials for Raspberry Pi;, n.d.)

Networking: The devices communicate mainly through Wi-Fi, allowing for seamless communication between the Raspberry Pi, the server and the user interface. This is to ensure real-time data transmission and alerts, assisting remote monitoring of conditions and notifications.

Protocols Used: MQTT operates on a broker-client model, where the broker manages message distribution between clients. In this system, the Raspberry Pi functions as both an MQTT broker and client, handling data from various sensors and distributing it to subscribed devices or applications.

Data Transmission: In this system, data transmission occurs over a Wi-Fi network using the MQTT protocol, where the Raspberry Pi acts as the broker to manage the flow of information between devices. Data, such as sensor readings and alerts, is published to specific topics and securely transmitted to subscribers, like the web app or email server. To protect sensitive information, data packets are encrypted using Transport Layer Security (TLS), and authenticated access ensures that only authorized devices can participate in the communication.

User Interaction (UI/UX Design)



//Template taken from w3schools https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_analytics&stacked-h

Temperature	Humidity	Gas Level	Light level
8.0°C	70.0°C	Safe	376°C

Alerts

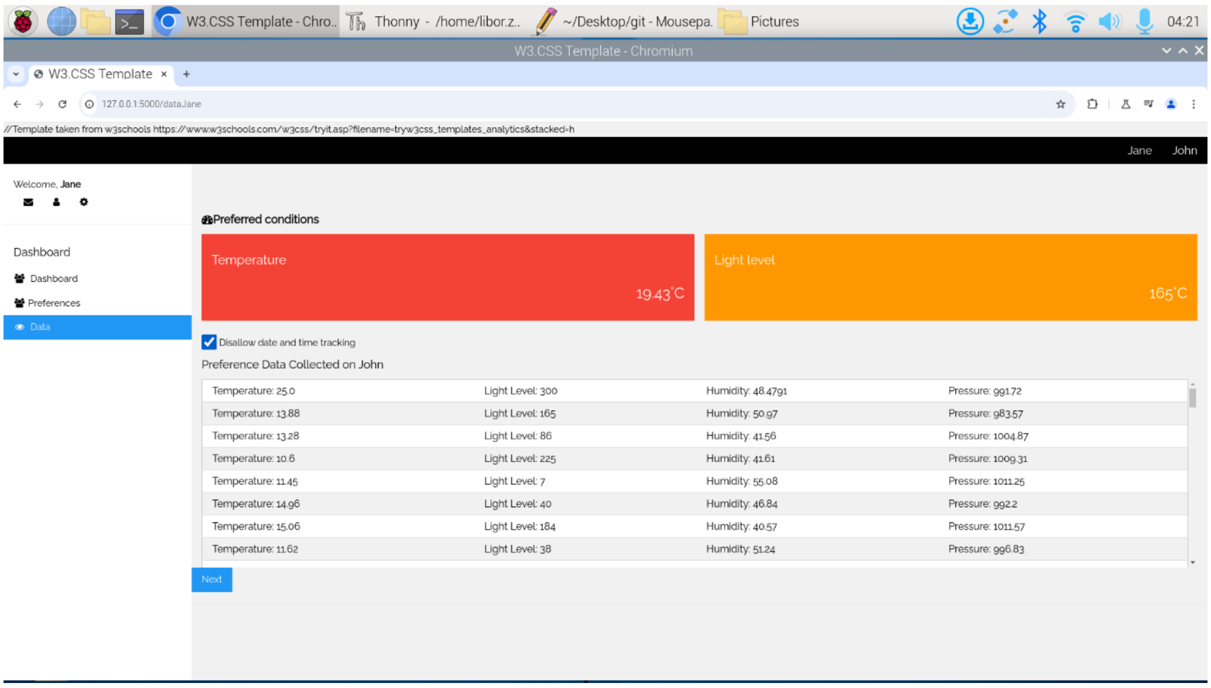
🔔	Gas Detected!	1:00 - 3/12/2024
🔔	Gas Detected!	21:00 - 14/11/2024
🔔	Gas Detected!	17:00 - 10/11/2024
🔔	Gas Detected!	5:00 - 8/11/2024
🔔	Gas Detected!	1:00 - 18/10/2024
🔔	Gas Detected!	1:00 - 6/10/2024
🔔	Gas Detected!	21:00 - 18/9/2024
🔔	Gas Detected!	21:00 - 12/9/2024
🔔	Gas Detected!	13:00 - 20/8/2024

Temperatures Recorded Today

Temperature: 18.89	18:00 - 30/11/2024
Temperature: 13.88	15:00 - 30/11/2024
Temperature: 4.38	14:00 - 30/11/2024

Figure 5: Example Dashboard

The website opens up the user Dashboard as shown in Fig. 5. The first thing the user sees is the general stats for the current day. At the top, the user can see colour coded boxes displaying the last readings taken from the sensors which show the temperature, humidity, light level and whether the gas level is safe or unsafe. Beneath those, the user can see a list containing all the gas and fire alerts that have been detected. These alerts can also be seen on the user's email as alerts are sent as soon as fire or gas have been detected. On the dashboard, the user can also see four tables, each displaying all the reading taken that current day. In order, the tables show: temperature, light level, humidity and pressure. Beneath the tables, the user can see four graphs showing historical data including the average reading for each statistic each month and year. This allows the user to easily identify trends in energy use. The black bar located at the top of the screen allows the users to switch profiles by clicking on each name. When the user clicks on 'Jane' from any page, it will reload the page and show information pertaining to Jane. The side bar at the side of the screen allows the user to switch between the pages.


//Template taken from w3schools https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_analytics&stacked=h

Welcome, **Jane**

Preferred conditions

Temperature 19.43˚C

Light level 165˚C

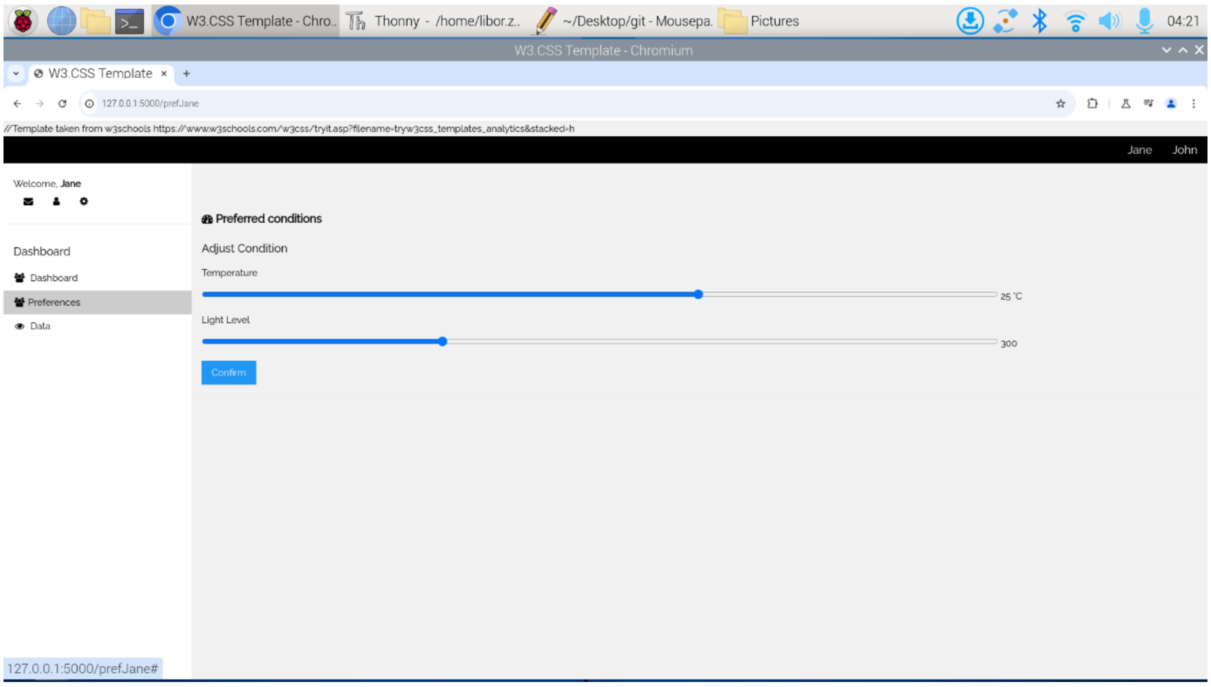
☑ Disallow date and time tracking

Preference Data Collected on John

Temperature: 25.0	Light Level: 300	Humidity: 48.4791	Pressure: 991.72
Temperature: 13.88	Light Level: 165	Humidity: 50.97	Pressure: 983.57
Temperature: 13.28	Light Level: 86	Humidity: 41.56	Pressure: 1004.87
Temperature: 10.6	Light Level: 225	Humidity: 41.61	Pressure: 1009.31
Temperature: 11.45	Light Level: 7	Humidity: 55.08	Pressure: 1011.25
Temperature: 14.96	Light Level: 40	Humidity: 46.84	Pressure: 992.2
Temperature: 15.06	Light Level: 184	Humidity: 40.57	Pressure: 1011.57
Temperature: 11.62	Light Level: 38	Humidity: 51.24	Pressure: 996.83

Figure 6: Data (Checked)

When the user clicks on 'Data' in the sidebar as shown in Fig. 6, they are taken to a personal page showing the user all the data collected on them. The two colourful boxes on top show the user the preferred conditions, including temperature and light level, which was determined by a machine learning algorithm analysing the users historical choices and the current date. Beneath the boxes, the user can see all the data recorded on the user, such as temperature, light level, pressure, humidity, time and date. The data is sorted into multiple pages, with a limit of a hundred records shown on one page. Only the hour is saved and displayed, the minutes are not recorded to help protect the user from having their schedule easily identified. The user can use the 'Next' and 'Previous' buttons to look through all the data. A checkbox above this data allows the user to choose to opt-out of having time and date collected, in effort to protect their privacy.


//Template taken from w3schools https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_analytics&stacked=h

Welcome, **Jane**

Preferred conditions

Adjust Condition

Temperature 25 ˚C

Light Level 300

Figure 7: Preference

When the user clicks on ‘Preferences’ as shown in Fig. 7 in the side bar, they can set the temperature and light level of their home, even when away from home. The user can set the temperature and light level by using the sliders. The ‘Confirm’ button then sends the data to the database. The user can then return to ‘Data’ and see the new record in their data list. By clicking ‘Dashboard’, the user can return to the main dashboard page.

Implementation

Development Process

Sensor Integration and Calibration - The first stage involved integrating the Sense HAT sensors with the Raspberry Pi and developing calibration techniques to ensure accurate data. This included recalibrating the temperature sensor to account for the heat generated by the Raspberry Pi itself.

Database Creation and Insertion - A MySQL database was set up to store sensor data, user preferences, and alerts. This stage also included implementing efficient data insertion methods through MQTT communication, ensuring real-time updates to the database.

Alert System Development - The system was programmed to detect hazardous conditions, such as gas leaks or fires, and provide visual alerts using RGB LEDs. An email alert system was also developed, allowing users to stay informed about their home’s safety remotely (Random Nerd Tutorials, n.d.).

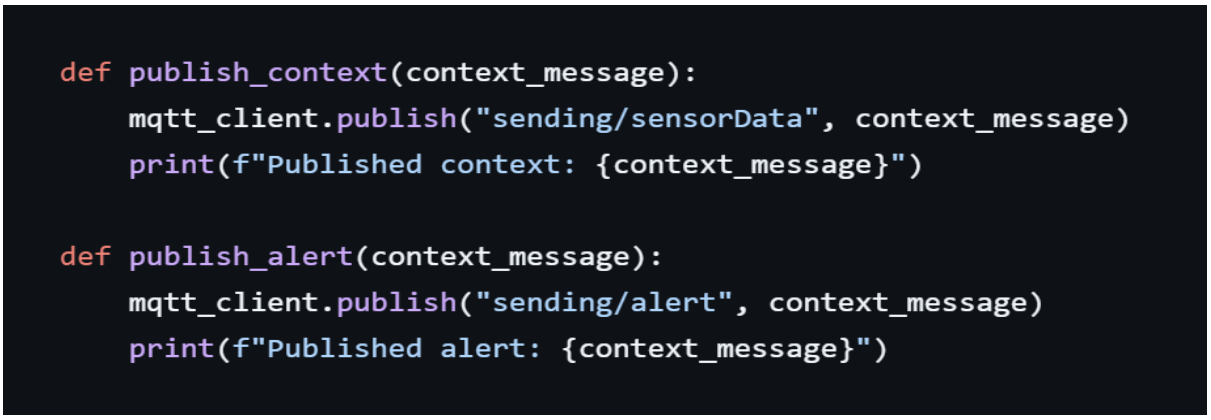
AI Algorithm Development – Various machine learning algorithms, including Decision Trees and Random Forest, were tested for accuracy and computational efficiency. Random Forest was selected and deployed on the Raspberry Pi for real-time predictions of optimal temperature and light intensity.

User Interface – A web dashboard was developed for remote monitoring, along with RGB LED-based visual alerts for local feedback. The interface allows users to view sensor data and receive alerts in real time. It also displays graphs based on historical data taken from the users and from the sensors. The webapp allows for a change in users which alters the information shown. The Dashboard page displays the current house conditions, all historical alerts, data taken from the current day, and the graphs. The Data page displays all the data taken from the current user and allows the user to remove sensitive data such as time and dates, which can help predict the user’s movements and schedule.

System Integration and Testing - All components were integrated and tested in simulated scenarios due to limitation on the sensors, but this is to ensure functionality, accuracy, and responsiveness in case a situation similar to the simulation arises.

Key Code Snippets and Algorithms

Code Highlights:

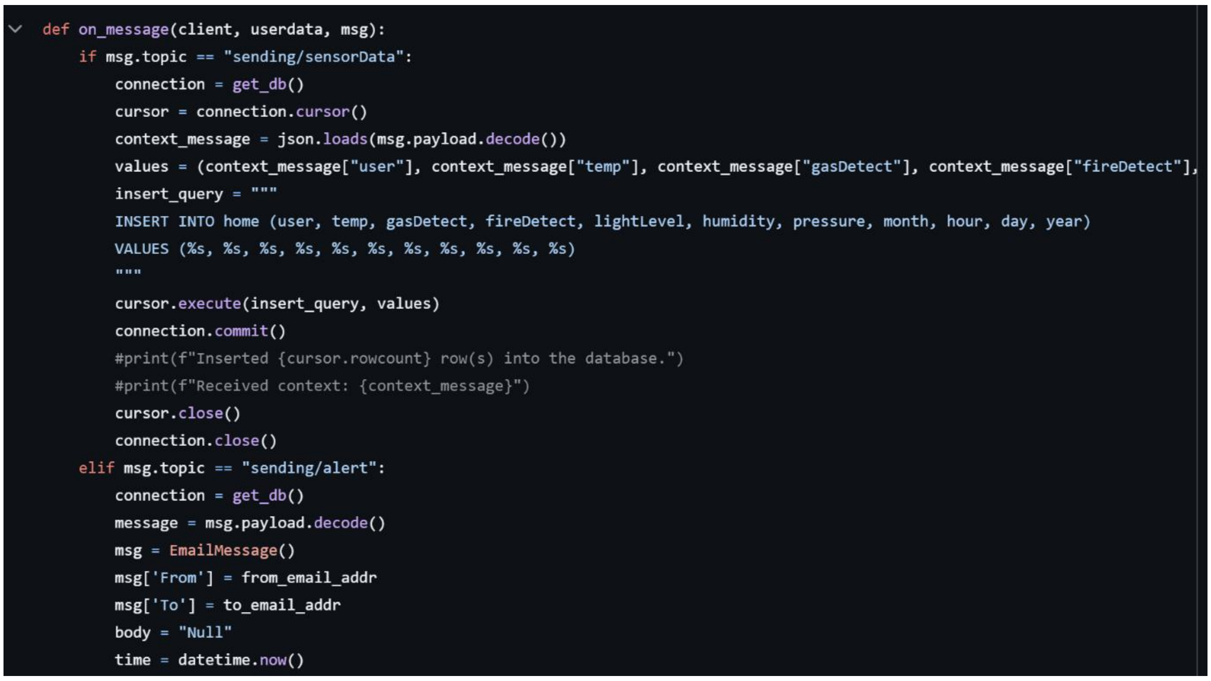


```python
def publish_context(context_message):
    mqtt_client.publish("sending/sensorData", context_message)
    print(f"Published context: {context_message}")

def publish_alert(context_message):
    mqtt_client.publish("sending/alert", context_message)
    print(f"Published alert: {context_message}")
```

Purpose: These functions send sensor data and hazard alerts over the MQTT protocol.

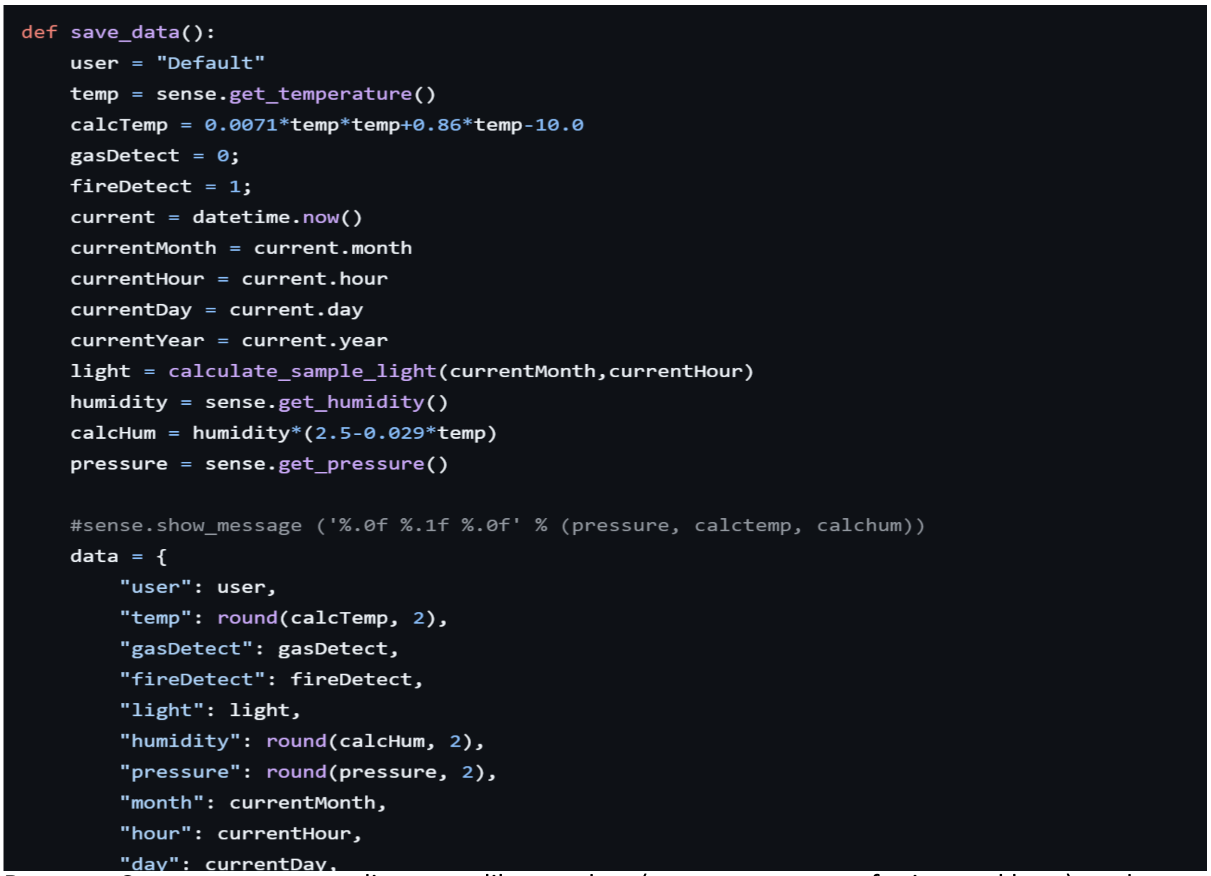
Significance: Enables real-time communication between sensors and the central processing system.



```python
def on_message(client, userdata, msg):
    if msg.topic == "sending/sensorData":
        connection = get_db()
        cursor = connection.cursor()
        context_message = json.loads(msg.payload.decode())
        values = (context_message["user"], context_message["temp"], context_message["gasDetect"], context_message["fireDetect"],
        insert_query = """
        INSERT INTO home (user, temp, gasDetect, fireDetect, lightLevel, humidity, pressure, month, hour, day, year)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
        """
        cursor.execute(insert_query, values)
        connection.commit()
        #print(f"Inserted {cursor.rowcount} row(s) into the database.")
        #print(f"Received context: {context_message}")
        cursor.close()
        connection.close()
    elif msg.topic == "sending/alert":
        connection = get_db()
        message = msg.payload.decode()
        msg = EmailMessage()
        msg['From'] = from_email_addr
        msg['To'] = to_email_addr
        body = "Null"
        time = datetime.now()
```

Purpose: Sends hazard alerts (e.g., fire or gas) to a predefined email address.

Significance: Provides remote monitoring capability for users.

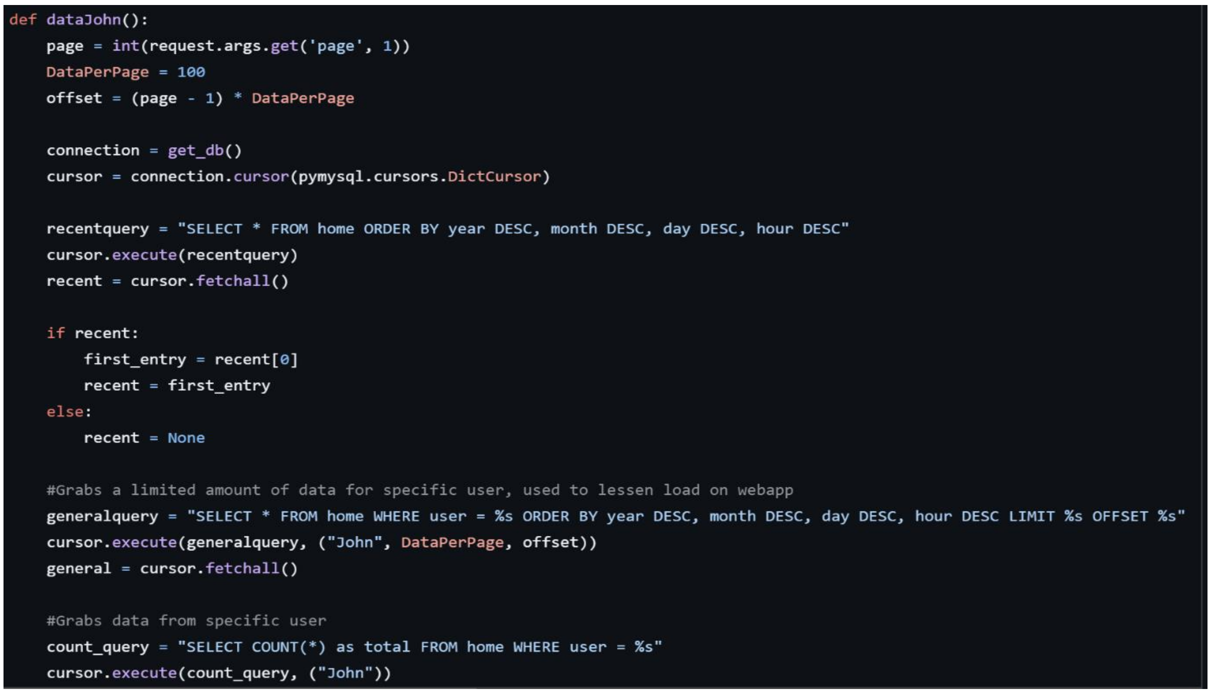


```python
def save_data():
    user = "Default"
    temp = sense.get_temperature()
    calcTemp = 0.0071*temp*temp+0.86*temp-10.0
    gasDetect = 0;
    fireDetect = 1;
    current = datetime.now()
    currentMonth = current.month
    currentHour = current.hour
    currentDay = current.day
    currentYear = current.year
    light = calculate_sample_light(currentMonth,currentHour)
    humidity = sense.get_humidity()
    calcHum = humidity*(2.5-0.029*temp)
    pressure = sense.get_pressure()

    #sense.show_message ('%.0f %.1f %.0f' % (pressure, calctemp, calchum))
    data = {
        "user": user,
        "temp": round(calcTemp, 2),
        "gasDetect": gasDetect,
        "fireDetect": fireDetect,
        "light": light,
        "humidity": round(calcHum, 2),
        "pressure": round(pressure, 2),
        "month": currentMonth,
        "hour": currentHour,
        "day": currentDay,
```

Purpose: Captures sensor readings, recalibrates data (e.g., compensates for internal heat), and structures it into JSON format.

Significance: Ensures accurate data collection and standardized transmission to the database.



```python
def dataJohn():
    page = int(request.args.get('page', 1))
    DataPerPage = 100
    offset = (page - 1) * DataPerPage

    connection = get_db()
    cursor = connection.cursor(pymysql.cursors.DictCursor)

    recentquery = "SELECT * FROM home ORDER BY year DESC, month DESC, day DESC, hour DESC"
    cursor.execute(recentquery)
    recent = cursor.fetchall()

    if recent:
        first_entry = recent[0]
        recent = first_entry
    else:
        recent = None

    #Grabs a limited amount of data for specific user, used to lessen load on webapp
    generalquery = "SELECT * FROM home WHERE user = %s ORDER BY year DESC, month DESC, day DESC, hour DESC LIMIT %s OFFSET %s"
    cursor.execute(generalquery, ("John", DataPerPage, offset))
    general = cursor.fetchall()

    #Grabs data from specific user
    count_query = "SELECT COUNT(*) as total FROM home WHERE user = %s"
    cursor.execute(count_query, ("John"))
```

Purpose: Retrieves and displays recent sensor data, hazard alerts, and monthly averages on the user dashboard.

Significance: Offers users a clear and interactive interface to monitor their home

```python
# Integrated Machine Learning Model
def initialize_ml_models():
    connection = get_db()

    query = "SELECT * FROM home WHERE user = 'John' ORDER BY year DESC, month DESC, day DESC, hour DESC"
    data = pd.read_sql(query, connection)

    connection.close()

    X = data[['user', 'hour', 'month', 'gasDetect', 'fireDetect', 'day', 'year', 'humidity', 'pressure']]
    X = pd.get_dummies(X, columns=['user'], drop_first=True)
    y_temp = data['temp']
    y_light = data['lightLevel']

    X_train, X_test, y_temp_train, y_temp_test, y_light_train, y_light_test = train_test_split(
        X, y_temp, y_light, test_size=0.2, random_state=42
    )

    temp_model = RandomForestRegressor(random_state=42)
    light_model = RandomForestRegressor(random_state=42)

    temp_model.fit(X_train, y_temp_train)
    light_model.fit(X_train, y_light_train)
```

Purpose: Initializes and trains two Random Forest models to predict temperature and light preferences based on historical data.

Relevance: This core functionality adapts the smart home environment to user-specific preferences dynamically.

```python
def run_predictions():
    time = datetime.now()
    current_conditions = {
        'user': "John", 'hour': time.hour, 'month': time.month, 'day': time.day, 'year': time.year,
        'gasDetect': 0, 'fireDetect': 0, 'humidity': 45, 'pressure': 100
    }
    input_data = pd.DataFrame([current_conditions])
    input_data = pd.get_dummies(input_data, columns=['user'], drop_first=True)

    missing_cols = set(X.columns) - set(input_data.columns)
    for col in missing_cols:
        input_data[col] = 0
    input_data = input_data[X.columns]

    predicted_temp = temp_model.predict(input_data)[0]
    predicted_light = light_model.predict(input_data)[0]

    return predicted_temp, predicted_light
```

Purpose: Uses Random Forest models to predict optimal environmental conditions based on current data.

Significance: Adapts the home environment dynamically to user needs.

```python
def calculate_monthly_averages(data):
    monthly_data = {} #Defines a dictionary

    for entry in data:
        year = entry.get('year')
        month = entry.get('month')


        if year not in monthly_data:
            monthly_data[year] = {}
        if month not in monthly_data[year]:
            monthly_data[year][month] = {
                'temp': [],
                'humidity': [],
                'pressure': [],
                'lightLevel': []
            }
        monthly_data[year][month]['temp'].append(entry['temp'])
        monthly_data[year][month]['humidity'].append(entry['humidity'])
        monthly_data[year][month]['pressure'].append(entry['pressure'])
        monthly_data[year][month]['lightLevel'].append(entry['lightLevel'])
```

Purpose: Aggregates stored sensor data to compute monthly averages for temperature, humidity, and light levels.

Significance: Provides insights into long-term environmental trends for users to optimize energy consumption.

Algorithm Explanation:

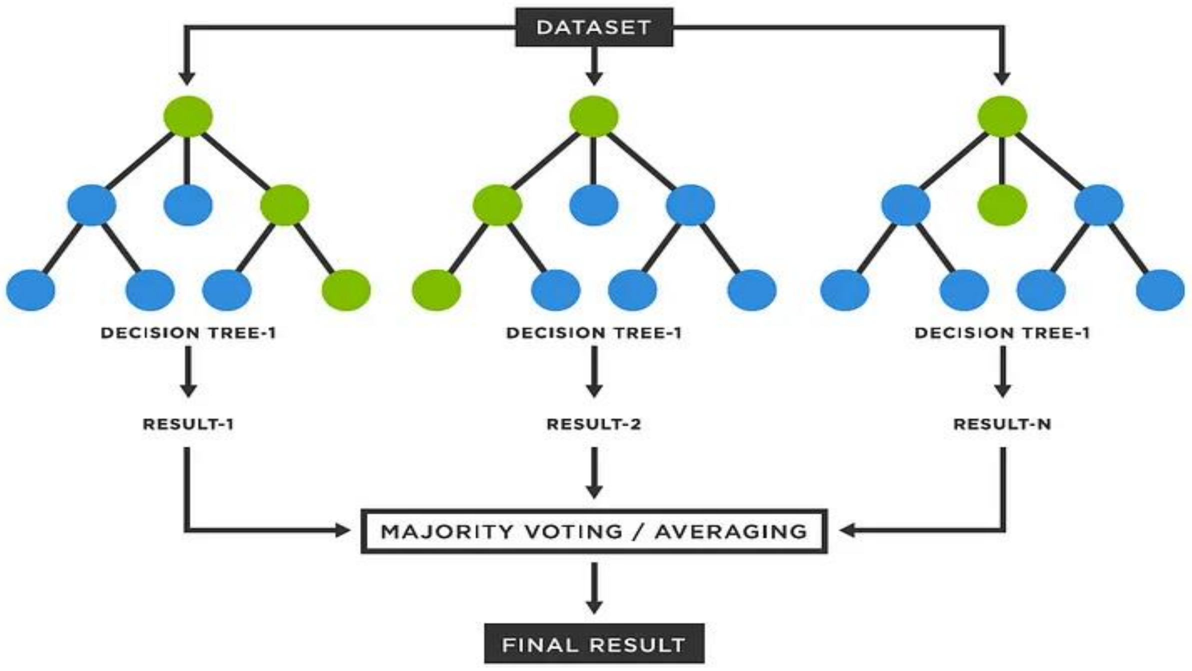


Figure 8: Random Forest

The algorithm applied to the project, in particular, the Random Forest algorithm has a significant importance for tailoring the custom user context and guaranteeing that hazards and detected properly. As the algorithm operates, multiple decision trees are developed during the training phase and their predictions are combined to enhance precision as well as strength as shown in Fig. 8 (Alam, 2024) . Each tree independently evaluates a subset of the input data, such as temperature and their preferences and provides a prediction. These individual predictions are then averaged (for regression) to provide the ultimate answer. This method lowers the chances of overfitting, ensuring that the model performs adequately even in the presence of random or noisy data, which in turn makes it suitable for practical implementation such as on a Raspberry Pi.

The algorithm is trained on a data that was pre-recorded from sensors and user inputs to make the prediction of two important factors: temperature and light intensity. These predictions enable the system to make adaptations dynamically, providing comfort and make it energy efficient. For example, based on the user's preference and the condition of the environment, the algorithm suggests an optimal setting that align with personal comfort while minimizing energy consumption. Moreover, the application of the algorithm also includes hazard detection, also, since the sensors are also calibrated through the system, it helps detect abnormal temperature spikes which may indicate fire. Real time decision making can also be achieved as the model is run locally on the Pi without relying on external servers, enhancing the system's privacy and also reliability. The balance of simplicity, accuracy and computational efficiency of Random Forrest makes it a fundamental component in achieving the system's goal of smarter, safer home automation.

Performance Metrics: The evaluation of the model involves the use of R-squared ratings and mean squared error calculations. These calculations determine how accurate the prediction of the model is. The model, for temperature, scored a Mean Squared Error (MSE) of (2.33) and Coefficient of Determination (R^2) of (0.88), which is a high precision measure with a strong linear correlation of the prediction to the actual values. And for light intensity, the MSE = (5114.60) R^2 = (0.85). These results indicate the algorithm is efficient in dealing with non-linear problems of a high complexity while remaining relatively efficient computationally on Raspberry Pi. This performance guarantee early enough detection and correction of danger.

Tools, Libraries, and Frameworks Used

Programming Languages:

Python: The system was developed in python language primarily because it encompasses sensor data collection and allows the incorporation of a machine learning model.

HTML: The user interface/web app interface was created in conjunction with HTML that allows graphs, alerts, and other such items to be presented neatly and user centred.

SQL: Enables users to control databases by managing, storing, and fetching the sensor information and their preferences.

Libraries and Frameworks:

Scikit-Learn: Employed in the design and development of the algorithm, more specifically the Random Forrest that was used in predicting the user's preferences.

Pandas: Utilized for manipulating and pre-processing data such as cleaning and converting the data into the formats that can be used for analysis.

Flask: This is a framework for Python that was integrated to design the backend of the web application enabling the generation of dynamic content and interaction with the target users.

Numpy: Used to perform numerical calculations and to efficiently work with the given dataset.

Paho-MQTT: This was used in the implementation of the MQTT protocol to facilitate the transmission of information between the pi and other systems.

Sense-Hat Library: It was used in collecting statistics and in creating animations for Led displays on the Sense-Hat.

Chat-GPT was used minimally for assistance in syntax errors as well as in a function in pref.html and prefJane.html. This was generated with the prompt "How to transfer values from html to python script" and helped learn how to effectively pass the learned preferences back to the script so it can be added to the database. Also, it was used to help summarize and make some grammatical corrections and better wording on some of the paragraphs. It was for suggestions mainly for User Manual and Installation Guides. The prompt used was "Give some suggestions for manual and installation of this system".

Hardware Components:

Raspberry Pi: This is the central processing unit where the sensors reside, the algorithm runs, and the local data processing takes place.

Sense-Hat: Consists of temperature, humidity and pressure sensors as well as an RGB Led matrix used for providing a visual alert.

Wi-Fi Module: Allows the interaction of the Pi with other devices by means of the MQTT protocol.

Challenges and Solutions

Technical Challenges: There were several challenges in the course of developing this project. One of the major issues was improper readings from the Sense-Hat's temperature sensor. The sensor was often more ready to record high temperature because the Pi from which it took readings was located directly above the CPU which generated extra heat. This prompted the need to formulate a recalibration equation to compensation on the demand sensor reading changes. Also, the processing constraints of the Pi in itself was also a difficulty, particularly in regard to the efficient algorithms that were supposed to be run in real time. The computation and the power next to the performance the system had to manage the constrain on the optimality. One more challenge was regarding how best to secure transmission of data over the MQTT protocol. Ensuring communication was secure on login with TLS over transport layer security encryption assist ease transition is essential use of this protocol is ideal for Internet of Things, however, is light. Further, we were not able to achieve detection of fire and gas leaks since we did not have access to the requisite fire and gas leak sensors which made us rely on preset conditions which could lead to more risk of false detection. Finally, the user interface development of the various users.

Such circumstance was a trial on its own, as it also required efforts of creativity in problem solving as well as the effective management of resources to obtain a useful and reliable system

Problem-Solving Strategies: With problem comes solutions. To address the challenges faced during the development of this system, a number of strategies was used. For the inaccurate reading of the sensor data, we implemented temperature recalibration formula (Bailey, 2018) to exclude the heat that was generated by the Pi. For example, an equation was used

Lessons Learned: This project furthered our knowledge in the social, technical and practical aspects that revolve around building the project. One of the most important things we learned is that IoT applications are only as reliable as their sensors, which should be reliably calibrated. When the system is built to take inputs for systems that are required for the detection and mitigation of hazards, the accuracy of the readings is critical for the calibration. In addition, we developed an awareness of edge computing problems, particularly the fact that deep learning techniques need to be tailored for resource-constrained devices such as a Raspberry Pi. Viewing things from a user perspective, we understood the need to focus on designing user-friendly interfaces that are more easily navigable by different users including the elderly and other such groups. Other issues such as offering data within the visually impaired population and issues related to privacy were also consistent with the need to respect functionality but also have a degree of inclusiveness and trust. In addition, this project acted as a great reminder to the need for an interdisciplinary approach, since hardware, software and machine learning integration, all of them required a tripartite approach. Finally, the issues that have to do with ethics mainly the questions of data privacy and sustainable development underscored the necessity for appropriate technological development and enhanced our resolve to outline every initiative in terms of energy saving and openness.

Ethical and Societal Considerations

Data Security and Privacy

Data Handling: User information is obtained from the Sense HAT sensors assembled with the Raspberry Pi onboard temperature, humidity and pressure parameters. In order to achieve higher privacy and security the data is processed at edge devices instead of third-party servers. Data is stored in a MySQL database and contains data such as user surname, sensor readings, and general timestamps for hours and dates. For the sake of user privacy, all the exact minutes are omitted because such information can give hackers a complete schedule of a particular user. Visualized outcomes are then displayed on a web dashboard owned by a certain user allowing easy supervision and exam of the information.

Privacy Measures: To safeguard the confidential information, several privacy mechanisms are in place:

Encryption: any communication is secured via STARTTLS encryption so that only email alerts for risks, e.g., danger from fire or gas leakage, are sent to recipients without the interference of outsiders.

Data Anonymization: Only first name of the users is stored in the database, other identification information such as their last names is either deleted or not collected so that the opportunities of re-identification are lessened.

Local Data Processing: Raspberry Pi performs all the required data processing on the sensor data before the information is sent over the network or is stored to ensure safety from foreign networks.

User Consent: The system allows users to retain complete control over their information and aims to give users order of transparency:

Dashboard Features: Users have control over all the information that is collected from them including environmental information along with timestamps which helps them verify the contents stored on the system.

Data Removal Options: Users can request to permanently delete sensitive information such as timestamps from the web app. This further enhances user privacy as the deletion feature is anticipated to achieve such a goal without rendering the system undesirable.

Informed Consent: Users are explained through the interface of the web app on the type and the purpose of the data that has been sourced. The system is forthright on the manner in which such data is utilized for making predictions, monitoring and sending alerts, therefore building trust.

The system is centred around encryption, user control, and anonymization which strengthen privacy and security of the user and compliment the ethical concerns of smart home use case scenarios.

Legal Implications

The system has been constructed while keeping key legal requirements in mind to protect the personally identifiable information and all the data protection laws such as General Data Protection Regulation (GDPR). The system implements the following principles in order to comply with GDPR:

Data Minimization: Only first names and environmental readings are collected and stored if they are necessary.

Transparency: Users are given authority over the data collected by telling them exactly what information is recorded and how it is utilized.

Right to Erasure: This design enables users of the web app to remove such sensitive information such as timestamps thereby complying with the GDPR “right to be forgotten” directive.

The design further deploys measures such as local processing within Raspberry Pi which ensures that external servers do not have the opportunity to engage with the devices independently, ensuring compliance with data privacy requirements.

The Enforcement application has some potential legal liabilities which are mainly associated with data breaches, false alerts or incorrect updates and system failures. For example:

Data Breaches: Data transmission can be protected using adequate encryption such as STARTTLS for email alerts and TLS for MQTT communication. However, if there is data breach in strategy, sensitive data is exposed leading to legal ramifications.

False Alerts: The use of incorrect hazard detection in the systems where false warnings of fire or gas leak are provided could cause users to go through unwarranted panic or pain. In order to mitigate this issue stern disclaimers have been placed to protect users from the boundaries of the system.

System Failures: Liability may arise where the system fails to detect an actual hazard, and this does not get reported by the system at all. In order to address this issue, the system has been configured in such a manner that data is processed at the site and quick response is given.

Societal Impact and Accessibility

Accessibility: The system is designed to be easier to use for a wider variety of users. This includes allowing for the elderly, and people with disabilities to make use of the smart home safety system. Visual hazard notifications in the form of animated LED alerts are issued through colour coding, which makes the system easier to use for users who cannot hear. The web dashboard is designed in such a way that it is easy to navigate, and the data is easy to visualize, making the interface user friendly even to low technostress users. Users are also free to adjust their settings according to their preferences, while certain privacy-sensitive information can also be removed from being collected altogether. Therefore, these features make sure that accessibility is met but systems do not become unsafe or unusable.

Ethical Use: The devices making up the smart home security system are designed to protect user privacy in an open, inclusive and ethically sound manner. Protection of privacy is one of the main objectives of the application since all sensor data is processed on the Raspberry Pi without the need to use external servers. This greatly reduces the risks of illegal data access or theft and makes the processing of information more secure. It is also made very clear to users what the purposes of the collection are, and what information is gathered, that is, only first names and general time stamps. Users can also erase sensitive information such as specific time and date and thus gain complete control over their information. Inclusivity represents another ethical aspect of such systems. Such customization facilities as visual hazard indication together with a simple web interface make the system user friendly for a large number of people, including older adults and the disabled. Furthermore, this system also enhances transparency among users by depicting both the capabilities and the limitations of the system in order to build trustworthy relationships and promote responsible choices.

Sustainability: With a view to sustainability, the system is directed at minimizing environmental degradation and increasing energy efficiency. The central processing module is the Raspberry Pi, which is energy-efficient and optimizes energy consuming machine learning tasks – operating on the edge. Reducing reliance on cloud computing by executing a significant amount of tasks at the edge leads to a decrease in the overall carbon footprint due to the reduction in energy consumption. Moreover, durable components, such as the Sense HAT, are used that economize electronic waste by ensuring that the system can be used for a significantly longer period of time. Furthermore, the system provides users with graphical representations of energy trends over a defined period enabling users to use the historical data responsibly to alter the home heating, lighting, and other appliances. These features not only assist users in managing their energy consumption but also contribute to the accomplishment of greater sustainability objectives, thereby enhancing the application's reputation as an ecologically friendly option for smart home technologies.

Conclusion and Reflection

This project achieved creating a working prototype of a smart home safety system which combines real-time environment monitoring, user-centric adjustments, and hazard detection. Key accomplishments include building a machine learning model through Random Forest capable of predicting ideal temperature and light conditions for each user. The system using the Raspberry Pi as an edge computing device to process data solves the issue of heavy dependency on servers from outside. A web dashboard interfacing was also developed so that the users can comfortably check the home setup and can survey over time while receiving notifications as required. The use of RGB LEDs together with email message alerts facilitates communication with the user and is aimed especially at the more vulnerable individuals.

The project reached its goals by focusing on reliable sensor calibration, systematic data preservation, and effective machine learning models. Data protection standards, such as those set by the GDPR, are satisfied through the adoption of data encryption, privacy features, reduced data collection, and opt-out choices. The system also successfully managed to fulfil its objective of conveying useful insights on historical data exploration by giving users the capability to control their homes in an optimal manner. The strong coupling of the components and their attributes also clearly suggests that the project objectives were not only satisfied but were also enhanced in terms of reach and human-centred design as well.

While the system achieved its primary objectives, it would be reasonable to note that there is scope for improvement. For instance, the inclusion of extra sensors in the setup, such as fire and gas sensors, would have improved the detection system by removing set conditions. Expanding the scope of interoperability to include smart appliances such as thermostats and dehumidifiers would improve the users' conveniences by allowing them to automatically control their environment. Most likely, a mobile app would offer users a wider scope in monitoring and controlling their homes. Last but not least, certification of safety and compliance can enhance the level of acceptance of the system in many different markets.

In the course of working on this project, we witnessed tremendous development in both our technical abilities and our capacity for collaborative problem solving. Engaging in the development of actual smart systems using IoT, machine learning and edge computing techniques gave us insights into the problems and scope in building them. As a group, we faced challenges like optimizing the system in view of the processing constraints and guaranteeing data security with a common dedication to delivering creativity. Our focus was directed toward mutual strengthening of hardware system integration, software development, and user interface design. The process of integrating machine learning models with edge devices, while building user-friendly interfaces, showcased the importance of interdependence and feedback in facilitating goal-based and project oriented task completion. While addressing accessibility and sustainability challenges, we further deepened our understanding of technology's social good when it is done responsibly. This has been one of the projects that have motivated us to seek further areas in which technology can add substantial value while ensuring that ethics and inclusivity are at the core of everything. It has also reinforced our belief in the ability to use modern solutions and frameworks and leaves us enthusiastic about the next steps in smart home systems and other areas.

References

Alam, M., 2024. *Understanding the Random Forest Algorithm – A Comprehensive Guide.* [Online]
Available at: https://datasciencedojo.com/blog/random-forest-algorithm/
[Accessed 10 December 2024].

Bailey, J., 2018. *Part 3. Sense HAT Temperature Correction.* [Online]
Available at: https://github.com/initialstate/wunderground-sensehat/wiki/Part-3.-Sense-HAT-Temperature-Correction
[Accessed 10 December 2024].

GeeksforGeeks, 2024. *Random Forest Algorithm in Machine Learning.* [Online]
Available at: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
[Accessed 10 December 2024].

Random Nerd Tutorials, n.d. *Raspberry Pi: Send an Email using Python (SMTP Server).* [Online]
Available at: https://randomnerdtutorials.com/raspberry-pi-send-email-python-smtp-server/#:~:text=You%20start%20by%20importing%20the,message%20module.&text=Next%2C%20you%20create%20variables%20for,email%20address%20to%20send%20to
[Accessed 10 December 2024].

Tutorials for Raspberry Pi;, n.d. *Wireless communication between Raspberry Pi's via MQTT broker/client.* [Online]
Available at: https://tutorials-raspberrypi.com/raspberry-pi-mqtt-broker-client-wireless-communication/
[Accessed 10 December 2024].

Appendices

Full Code Listings: All the code is available on the GitHub repository for reference: https://github.com/Izabela-Zelek/TheIlliterates

User Manuals:

1. Installation Guide

Step 1: Hardware Setup

1. For best coverage of the sensors, it is recommended that you locate the Raspberry Pi at the central location of the house.
2. Connect the Raspberry Pi GPIO pins to the Sense HAT.
3. Establish Wi-Fi connectivity and link the Raspberry Pi to an appropriate power source.
4. Make sure that the sensors (temperature, humidity, pressure) are not blocked in any manner for accurate readings to be taken.

Step 2: Software Setup

1. Connect the power chord to your Raspberry Pi and confirm if Raspbian OS has been installed.
2. Install required Python libraries. Guide on how to install libraries for python on Raspberry Pi is available here: https://projects.raspberrypi.org/en/projects/generic-python-installing-with-pip
3. Download the project files to your Raspberry Pi.
4. Start the MQTT client and Flask server: "python HomeManager.py"
5. Access the web app via the Raspberry Pi's IP address in a web browser (e.g., http://192.168.X.X:5000).

Step 3: Initial Configuration

1. Navigate to the web app dashboard and select your user profile (e.g., John, Jane).
2. Set the preferred temperature and light intensity settings via the preferences tab.
3. Ensure the email alert system is configured with the correct email address for receiving notifications.

2. User Guide

Dashboard Overview

The dashboard displays:

- Current environmental conditions (temperature, humidity, pressure).
- Historical trends with graphs for monthly averages.
- Recent hazard alerts (e.g., fire, gas).

Use the "Data" tab to view or delete stored data, including timestamps for privacy concerns.

Alert System

Visual Alerts: RGB LEDs on the Sense HAT display animations for hazards (e.g., red flames for fire).

Email Alerts: Ensure the system is connected to the internet to receive real-time notifications.

Data Management

View all collected data on the "Data" page.

Delete sensitive information, such as timestamps, using the "Remove Data" button for enhanced privacy.

3. Maintenance Tips

Periodically clean the Sense HAT to prevent dust accumulation on the sensors.

Check Wi-Fi connectivity regularly to ensure uninterrupted operation.

Update the system software and Python libraries for security patches and feature improvements.

4. Troubleshooting

No Data on Dashboard: Verify that the MQTT broker is running, and the Raspberry Pi is connected to Wi-Fi.

No Alerts: Ensure sensors are correctly connected and the calibration script is running.

Email Alerts Not Working: Check the SMTP configuration and internet connection.

This manual ensures smooth installation, operation, and maintenance of your smart home safety system. For further assistance, refer to the web app help section.